VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

POROVNÁNÍ CLOUDOVÝCH SYSTÉMŮ PRO SMART HOME

COMPARISON OF CLOUD SYSTEMS FOR SMART HOME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK SOBOL**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Sobol Patrik**
Program: Informační technologie
Název: **Porovnání cloudových systémů pro Smart Home**
Comparison of Cloud Systems for Smart Home
Kategorie: Softwarové inženýrství

Zadání:

1. Prostudujte problematiku cloudových systémů. Zaměřte se na služby Google a Amazon. Zaměřte se na služby pro realizaci smart home (komunikace se senzory, databáze, řídicí panel atd.).
2. Navrhněte vlastní řešení Smart Home s využitím existujících dostupných cloudovýh služeb Google a Amazon. Vyberte vhodné senzory a aktuátory pro smart home. Navrhněte simulátor smart home tak, aby bylo možné kombinovat reálné i simulované senzory a aktuátory a připojit je k cloudu.
3. Navržený systém realizujte tak, aby bylo možné alternativně využít služby Amazon a Google.
4. Proveďte testování a porovnejte obě řešení.

Literatura:

- Google cloud. URL: https://cloud.google.com/docs/
- Amazon Web Services. URL: https://docs.aws.amazon.com/index.html

Pro udělení zápočtu za první semestr je požadováno:

- První 2 body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Práca sa zaoberá porovnaním jedných z najväčích poskytovateľov cloudových služieb, Amazon a Google, s výrazným zameraním na služby urečené pre inteligentnú domácnosť. Skúma prepojenie dostupných služieb medzi sebou a zhŕňa aktuálne spôsoby prenosu dát medzi zariadením a cloudom za pomoci protokolu MQTT. V rámci práce je navrhnuté riešenie, kde kombináciou cloudových služieb vzniká ovládací panel, pomocou ktorého je možné ovládať zariadenia v inteligentnej domácnosti.

Abstract

This thesis deals with comparison of the biggest cloud computing platforms, Amazon and Google, with high interest in services focused on smart home. It examines connection of services between each other and it summarises options of data transport between device and cloud using protocol MQTT. By integrating of cloud services dashboard is created, which allows to control devices in household.

Kľúčové slová

Cloud, Cloudové systémy, Cloudové služby, Inteligentná domácnosť, Internet vecí, IoT, MQTT

Keywords

Cloud, Cloud systems, Cloud services, Smart Home, Internet of things, IoT, MQTT

Citácia

SOBOL, Patrik. *Porovnání cloudových systémů pro Smart Home*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Vladimír Janoušek, Ph.D.

Porovnání cloudových systémů pro Smart Home

Prehlásenie

Prehlasujem, že som túto bakalárksu prácu vypracoval samostatne pod vedením pána doc. Ing. Vladimíra Janouška Ph.D. Uviedol som všetky literárne prameňe, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Patrik Sobol
12. mája 2021

Poďakovanie

Týmto by som chcel poďakovať môjmu vedúcemu práce doc. Ing. Vladimírovi Janouškovi Ph.D. za trpezlivosť, ochotu, vecné pripomienky a rady, ktoré mi poskytol pri riešení tejto práce.

Obsah

Kapitola 1

Úvod

So slovným spojením Cloud computing alebo aj Cloudové služby sa dnes stretáva čoraz viac ľudí. Používajú ich nielen odborníci v IT sfére, ale aj bežní ľudia, ktorí si ani neuvedomujú, že s nimi pracujú. Cloudové služby idú do popredia výrazným tempom a otvárajú nové možnosti a pohľady ako riešiť zadané problémy. Jedná sa nielen o fyzické zariadenia, ale aj zdroje, či rôzne služby prevádzkované cloudovou spoločnosťou za poplatok. Mnoho novovytvorených firiem od začiatku svojho pôsobenia používa práve cloudové riešenie, ktoré je omnoho výhodnejšie ako si zabezpečiť celú infraštruktúru. Existujúce firmy na tento trend postupne nabiehajú, čím čiastočne presúvajú svoje systémy do cloudu. Tým vzniká model hybridného cloudu.

Táto práca sa zaoberá prieskumom a porovnaním dostupných cloudových služieb a ich prínos v kombinácii s inteligentnou domácnosťou.

Takmer každý z nás sa stretol so zariadením, ktoré spadá do oblasti inteligentnej domácnosti. Môže sa jednať o doplnky, ktoré rozširujú svoje bežné chovanie. Napríklad žiarovka, ktorá v minulosti len svietila, dnes môže meniť farbu, intenzitu žiarenia a byť ovládaná pomocou mobilnej aplikácie. Použitie môže byť skvelým príkladom toho, že bežní ľudia pracujú s cloudovými službami a ani o tom nevedia. Užívateľ má k dispozícii mobilnú alebo webovú aplikáciu, pomocou ktorej môže zariadenia v domácnosti ovládať. Komunikácia aplikácie so zariadeniami môže prebiehať prostredníctvom dostupných cloudových služieb na to určených. Napríklad, keď užívateľ obdrží upozorenie v mobile, že senzor teploty v domácnosti presiahol povolenú hodnotu, zariadenie najprv odošle správu do cloudu, kde pridelená služba vyhodnotí túto skutočnosť a odošle notifikáciu.

Tejto problematike sa chcem venovať práve preto, že ponúka možnosť ju použiť aj pri riešení v iných oblastiach. Nakoľko som o danej téme nemal dostatok informácií, videl som v nej možnosť ako nadobudnúť nové poznaky a skúsenosti. Porovnaním rôznych poskytovateľov cloudových služieb som získal prehľad nad dostupnými možnosťami, ktoré môžu uľahčiť rozhodnutie pri návrhu projektu.

Cieľom práce je priblížiť problematiku cloud computing a inteligentnej domácnosti čitateľovi, a tým mu ponúknuť rozhľad nad možnosťami implementácie.

V nasledujúcich kapitolách bude najskôr uvedenie do cloudu, inteligentnej domácnosti a jej zabezpečenia. Následne prieskum dostupných cloudových služieb a v poslednej časti bude predstavený môj návrh riešenia a implementácia komunikácie zariadení s cloudovými službami.

Kapitola 2

Cloud computing

Cloud a Cloud computing je nový model výpočtovej techniky, ktorý je integráciou pokročilých výpočtových modelov, sofistikovaných webových technológií a moderných technológií sieťovej komunikácie. [6]

Termín Cloud [10] označuje vo všeobecnosti priestor na internete, kde je možné ukladať všetky druhy informácií vrátane fotografií, hudby, dokumentov a videí, teda naozaj skoro všetko. Výhodou je, že je možné sa kedykoľvek a odkiaľkoľvek dostať k svojím dátam. Jedinou podmienkou je pripojenie k internetovej sieti.

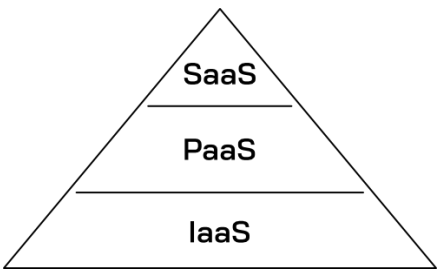
Ak hovoríme o Cloud computingu [10], hovoríme o poskytovaní služieb, aplikácií alebo programov uložených na serveroch na webe s tým, že používatelia k nim môžu pohodlne pristupovať napríklad pomocou webového prehliadača alebo klienta danej aplikácie a používať ich prakticky odkiaľkoľvek.

Výpočtový alebo sieťový prostriedok, aplikácia alebo akýkoľvek iný druh IT služby ponúkanej používateľovi prostredníctvom cloudu sa nazýva cloduová služba. Cloudové služby siahajú od jednoduchých aplikácií, ako sú e-mail a zdieľanie fotografií, až po výpočtové zdroje ponúkané ako služby hlavnými poskytovateľmi. [6]

S nárastom popularity Cloud computingu sa objavilo niekoľko rôznych modelov, ktoré pomáhajú uspokojovať špecifické potreby rôznych používateľov. Každý typ cloudovej služby poskytuje rôzne úrovne kontroly, flexibility a manažmentu.

2.1 Modely Cloud computingu

Existujú tri základné modely 2.1 pre Cloud computing. Každý model predstavuje inú časť cloud computingu. Okrem základných modelov existuje aj niekoľko ďalších. Každá kategória sa môže používať samostatne alebo v kombinácii s ostatnými.



Obr. 2.1: Schéma uvádza pohľad na tri základné modely služieb a ich nadväznosť.

Infraštruktúra ako služba

Infraštruktúra ako služba [8], z angl. Infrastructure as a Service, niekedy skrátene *IaaS*, obsahuje základné stavebné prvky cloudového IT a zvyčajne poskytuje prístup k sieťovým funkciám, virtuálnym počítačom a úložným priestorom. Ako užívateľ sa nestarám o konektivitu a bezpečnosť fyzických častí. Jedná sa o tú najnižšiu vrstvu, čo môže poskytovateľ ponúknuť svojim zákazníkom, ale najvyššiu úroveň flexibility a riadenia. Najviac sa podobá existujúcim zdrojom IT, ktoré dnes pozná mnoho IT oddelení a vývojárov.

Platforma ako služba

Platforma ako služba [8], z angl. Platform as a Service, skrátene *PaaS*, je model postavený nad *IaaS*. Odstraňuje potrebu spravovať základnú infraštruktúru a umožňuje sústrediť sa na správu aplikácií. Dobrým príkladom je databázový server. Zatiaľ čo pri *IaaS* by užívateľ musel server nainštalovať a popasovať sa s bezpečnostnými hrozbami, pri modeli *PaaS* sa poskytovateľ postará o infraštruktúru a užívateľ už pracuje len s finálnym produktom. Poskytovatelia týchto služieb presne vedia, čo databázaový server potrebuje pre správnu prevádzku. Model *PaaS* umožňuje vývojárom jednoducho vyvíjať aplikácie a implementovať ich bez priamej interakcie s podkladovou infraštruktúrou. Na modeli *PaaS* sú poskytované takmer všetky služby od Amazon Web Services a Google Cloud.

Softvér ako služba

Softvér ako služba [8], z angl. Software as a Service, niekedy skrátene *SaaS*, poskytuje výsledný produkt, ktorý prevádzkuje a spravuje poskytovateľ služieb. Vo väčšine prípadov ľudia, ktorí označujú softvér ako službu, označujú aplikácie pre koncových používateľov. Vďaka ponuke SaaS užívateľ nemusí premýšľať o tom, ako je služba udržiavaná alebo ako je spravovaná základná infraštruktúra. Musí len premýšľať o tom, ako bude používať konkrétny softvér. Bežným príkladom aplikácie *SaaS* je webový e-mail, kde môžete posielať a prijímať e-maily bez nutnosti udržiavať servery a operačné systémy, na ktorých je e-mailový program spustený.

2.2 Vlastnosti

Dôvodom prečo firmy používajú cloudové služby a nie vlastný hardware je rôznorodý.

Zdieľanie zdrojov

Je jednou z podstatných charakteristík Cloud computingu, kde poskytovateľ cloudových služieb môže zdieľať svoje zdroje medzi niekoľkých klientov a každému poskytnúť inú sadu služieb podľa jeho požiadaviek. Typickou stratégiou je poskytnutie virtuálneho počítača alebo služby na ukladanie dát [5].

Bezpečnosť

Fyzické servery sú uložené v prísne chránených datacentrách, ktoré musia spĺňať bezpečnostné kritéria [5]. Vstup ku serverom má len oprávnená osoba. Pravdepodobnosť krádeže fyzických zdrojov je takmer nulová. Ak by ale unikli dáta z produktu, nebude to chyba poskytovateľa, ale užívateľa, pretože pri svojom produkte použil nedostatočné zabezpečenie.

Dostupnosť

Užívateľ cloudových služieb môže pristupovať a spravovať produkt z rôznych kútov sveta. Stačí mu ku tomu len prístup na internet [5].

Môže nastať situácia, že prenos dát bude trvať dlhšie z dôvodu vzdialenosti od serveru. V tom prípade by mal poskytovateľ produktu porozmýšľať nad riešením, kde vo svete umiestniť ďalšie servery.

Škálovateľnosť a elasticita

Umožňuje užívateľom podľa potreby meniť výpočetné zdroje, tak aby sa zachoval rovnaký výkon pri zvýšení, resp. znížení záťaže. Užívateľ platí len za to čo reálne využíva [5]. Dobrým príkladom môže byť, že cez deň sú servery vyťažené omnoho viac ako cez noc. To umožňuje v noci niektoré servery vypnúť, poprípade cez deň ich spustiť viac.

Automatizácia

Je proces maximálneho využívania technológií automaticky inštalovať, konfigurovať a udržiavať cloudovú službu pri minimálnom manuálnom úsilí užívateľa. Automatizácia jednej služby vyžaduje nasadenie ďalších virtuálnych strojov, ktoré tiež vyžadujú neustálu údržbu [5].

Aktuálnosť

Poskytovateľ garantuje aktuálnosť hardvéru a softvéru. Užívateľ do procesu nemusí vôbec zasahovať [5].

2.3 Modely distribúcie Cloud computingu

Základné modely cloudových služieb sa zameriavajú na splnenie požiadaviek zákazníkov. Výber typu cloudovej služby má priamy vzťah s veľkosťou organizácie. Na základe distribúcie sú cloudy rozdelené do niekoľkých kategórií.

Verejený cloud

Cloudová aplikácia a všetky jej časti bežia v cloude [8]. Aplikácie v cloude boli buď vytvorené v cloude alebo boli presunuté z existujúcej infraštruktúry, aby sa využili výhody cloudového výpočtu. Je použiteľný na rôznych úrovniach abstrakcie. Verejný cloud používajú malé a stredné podniky ako svoj primárny výpočtový zdroj. Riešením môže byť aj pre veľké organizácie, ktoré nechcú investovať do vlastnej infraštruktúry, ale pociťujú prudký nárast dát na spracovanie.

Privátny cloud

Privátny cloud [8] vlastní a využíva organizácia výhradne pre svoje účely. Môžu ju k tomu viesť obavy spojené s bezpečnosťou a súkromím udajov verejných cloudov. Má plnú kontrolu nad cloudom, aplikáciami a údajmi. Na druhej strane musí znášať všetky náklady spojené s infraštruktúrou, údržbou a modernizáciou cloudu. Oproti verejnému cloudu je to nákladná služba, ktorú si môžu dovoliť iba veľké spoločnosti. Vo väčšine prípadov je tento

model distribúcie rovnaký ako pôvodná infraštruktúra, zatiaľ čo sa organizácia snaží využiť cloudové technológie.

Hybridný cloud

Hybridný model [8] distribúcie je spôsob prepojenia infraštruktúry a aplikácií medzi cloudovými prostriedkami a existujúcimi prostriedkami, ktoré sa v cloude nenachádzajú. Typicky sa jedná o kombináciu verejného a privátneho cloudu. Pre menej kritické služby sa využíva verejný cloud a kritické aplikácie organizácii sa spracuvávajú v privátom cloude. Hybridné cloudové služby poskytujú všetci hlavní poskytovatelia cloudových služieb.

Medzi najväčších poskytovateľov cloudových služieb patrí Amazon Web Services, Google Cloud, Microsoft Azure a DigitalOcean. Prvým dvom zmieneným venujem nasledujúce kapitoly.

Kapitola 3

Internet vecí

Internet vecí z angl. Internet of Things, skrátene *IoT*, je označenie pre pripojenie fyzických zariadení, vozidiel, domácich spotrebičov a ďalších zariadení k internetu. Zariadenia spolu komunikujú prostredníctvom siete a v prípade potreby si zdieľajú dáta a zasielajú správy, či signály alebo upozornenia [31]. Komunikácia prebieha vo väčšine prípadov bezdrôtovo pomocou *Wi-Fi* alebo *Bluetooth*.

Je ťažké presne definovať Internet vecí. Jednou z možných definícií je definícia podľa Webera: „Internet vecí je svet, kde sú fyzické objekty bezproblémovo integrované do informačnej siete, a kde sa fyzické objekty môžu stať aktívnymi účastníkmi v podnikových procesoch. K dispozícii sú služby k interakcii s týmito „inteligentnými objektami“ prostredníctvom Internetu. Je umožnené zisťovanie ich stavu a akýchkoľvek informácií spojených s nimi, berúc do úvahy problémy spojené s bezpečnosťou a súkromím“ [30].

Zariadenia v Internete vecí nazývame aj „inteligentnými“. Sú vybavené vstavanou mikroelektronikou, senzormi, rádiovými prijímačmi, softvérom, batériou alebo pripojením do elektrickej siete a do siete internetu pre vzájomnú komunikáciu medzi sebou [31]. Keby obyčajné predmety rozšírime o mikrokontrolér a pripojenie do internetovej siete, pomocou automatizácie im pridáme nový spôsob využitia, čím zlepšíme služby, ktoré práve poskytujú.

Internet vecí sa v súčasnosti teší veľkej prosperite a počet zariadení či systémov neustále rastie. Dnes ide najmä o zbieranie veľkého množstva dát v určitých časových intervaloch a ich následnej analýze.

3.1 Inteligentná domácnosť

Časť tejto práce je zameraná na podmnožinu Internetu vecí a to na Inteligentnú domácnosť. Názov je odvodený z ang. Smart Home, čo pomenúva domov, v ktorom je všetko pre komfort ľudí riadené technológiami. Špecifickým znakom pre tieto zariadenia sú procesor s menším výkonom, menšou pamäťou a zdrojmi napájania. Sú prispôsobené na nenáročné úlohy. Inteligentná domácnosť zahŕňa kontrolu osvetlenia, termoregulátora, klimatizácie, vypínačov, ale aj správu bielej techniky [24].

Zariadenia sú pripojené k centrálnej jednotke, tzv. gateway, ktorú vie užívateľ ovládať cez užívateľské rozhranie [24]. Niektoré zariadenia potrebujú podporný hub, pretože nemajú dostatočný výkon alebo technológiu na komunikáciu s centrálnou jednotkou. Hub vykonáva všetky dôležité operácie a zariadenia robia len to, na čo sú zamerané.

Pod centrálnou jednotkou sa rozumie spojenie hardvéru a softvéru, ktorý spracuváva prichádzajúce požiadavky, dáta a ovláda pripojené zariadenia. Spoločnosti, ktoré poskytujú

zariadenia môžu mať vytvorené vlastné riešenie centrálnej jednotky alebo využívajú riešenie cloudovej spoločnosti.

3.2 Komunikácia s centrálnou jednotkou

Ako bolo spomenuté vyššie, zariadenia komunikújú s centrálnou jednotkou bezdrôtovo. Najčastejšie prostredníctvom *Wi-Fi*, *Bluetooth* alebo *ZigBee*. Tým, že zariadenia, napr. videokamera, môžu v reálnom čase prenášať veľké množstvo dát, bolo nutné preskúmať nové možnosti akým spôsobom budú dáta prenášané tak, aby sa zachovala čo najväčšia kvalita pri najjednoduchšom prenosovom modeli. Najideálnejším protokolom na prenos dát zo zariadení inteligentnej domácnosti je protokol *MQTT*. Ale môžeme sa stretnúť aj s prípadmi, kedy sa používa protokol *HTTP* alebo *WebSockety*.

3.2.1 Protokol MQTT

Message Queuing Telemetry Transport (MQTT) je štandardný aplikačný protokol pre zasielanie správ medzi zariadeniami, ktorý využíva protokol TCP transportnej vrstvy a používa návrhový vzor publish-subscribe. Je navrhnutý ako jednoduchý a mimoriadne nenáročný na prenos správ a ideálny na pripojenie vzdialených zariadení, kde šírka pásma siete je obmedzená. Vďaka tejto nenáročnosti a jednoduchosti je ľahko implementovateľný do zariadení s menším výkonom procesoru. Dnes sa využíva v rôznych odvetviach priemyslu, ako napríklad automobilový priemysel, výroba, telekomunikácie, ropa a plyn [26].

Terminológia

Aplikačná správa - Údaje prenášané prostredníctvom MQTT protokolu po sieti. Pri prenose správa obsahuje dáta, Kvalitu služby (QoS), súbor vlastností a topic [26].

MQTT protokol definuje dva typy entít, *brokera* a *klienta*.

Klient - Akékoľvek zariadenie, od mikrokontroléra až po plnohodnotný server, ktoré používa knižnicu MQTT a je pripojené po sieti k serveru [26].

Server - Softvér bežiaci na počítači, resp. cloude, ktorý koná ako sprostredkovateľ medzi Klientom, ktorý publikoval aplikačnú správu a Klientom, ktorý sa prihlásil na odber tejto správy. Môžeme sa stretnúť aj s názvom broker alebo centrálny agent. Klienti medzi sebou nikdy nekomunikujú priamo, vždy využijú server [26].

Topic - Štítok, identifikátor, pripnutý k aplikačnej správe. Môže pozostávať z viacerých úrovní, ktoré sú oddelené lomkou [26].
Príkladom topicu, na ktorý sa môže klient prihlásiť je *bt/d-001/sensor/bme*, kde druhá časť *d-001* jednoznačne označuje číslo zariadenia. Použitím špeciálnych znakov je možné sa prihlásiť na celú skupinu topicov. Znak + nahrádza ktorýkoľvek popis jednej úrovne. Pri prihlásení na odber topicu *bt/+/sensor/bme* budú odberateľovi doručené správy zo všetkých zariadení bez ohľadu na ich identifikátor. Zástupný znak # nahrádza jednu alebo viac úrovní a musí byť uvedený vždy ako posledný. Pri prihlásení na odber topicu *bt/#* budú odberateľovi doručené správy zo všetkých zariadení, kde prvá úroveň topicu je *bt*.

Správy odoslané zo zariadení sú triedené podľa topicov. Klient buď publikuje správy v danom topicu, alebo je prihlásený na jeho odber. Jedno zariadenie môže byť v jednom okamihu publisher a zároveň aj subscriber.

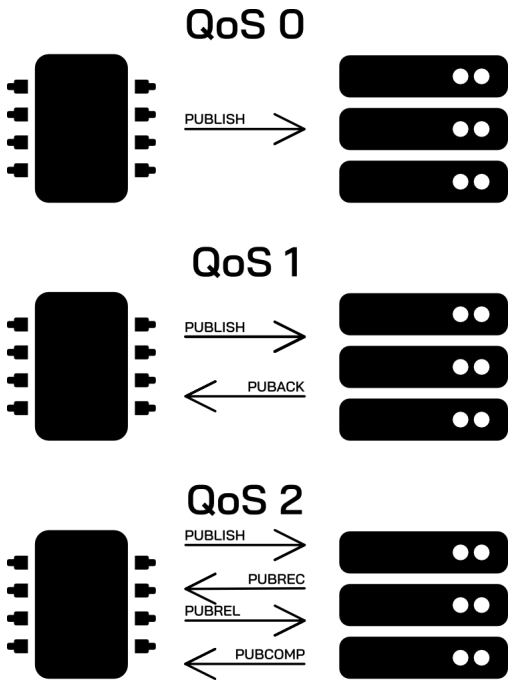
Správa typu Publish - Už ako je z názvu známe, slúži na publikovanie aplikačných správ zo zariadenia. Všetky tieto údaje zbiera MQTT Broker, ktorý ich rozosiela odberateľom [26].

Správa typu Subscribe - Udáva, že zariadenie žiada o prijímanie správ podľa zadaného topicu. Následne, každá takáto správa typu Publish prijatá do MQTT Brokera je odoslaná zariadeniu, ktoré sa prihlásilo na odber [26].

Kvalita služby - Skrátene QoS, definuje garanciu doručenia aplikačných správ. Môže nadobúdať tri úrovne, viď obr. 3.1, ktoré sa od seba odlišujú podľa toho ako je zaistené dodanie [26].

Úroveň 0 (najviac raz, fire and forget) udáva, že správa je odoslaná len raz. Klient a ani broker nevynakladajú ďalšie úsilie na potvrdenie doručenia.
Úroveň 1 (aspoň raz, acknowledged delivery) udáva, že správa je opakovane odosielaná pokým odosielateľ neobdrží potvrdenie o doručení.
Úroveň 2 (práve raz, assured delivery) spája odosielateľa a prijímateľa pomocou *two-level* handshake, čím zaručí prijatie práve jednej kópie správy.

Táto vlastnosť nemá vplyv na spracovávanie dát v TCP/IP transportnej vrstve. Je používaná výhradne medzi odosielateľmi a prijímateľmi správ v MQTT aplikačnej vrstve.



Obr. 3.1: Názorná ukážka zobrazujúca publikovanie správy medzi zariadením a brokerom pri rôznych úrovniach kvality služby.

Výmena správ

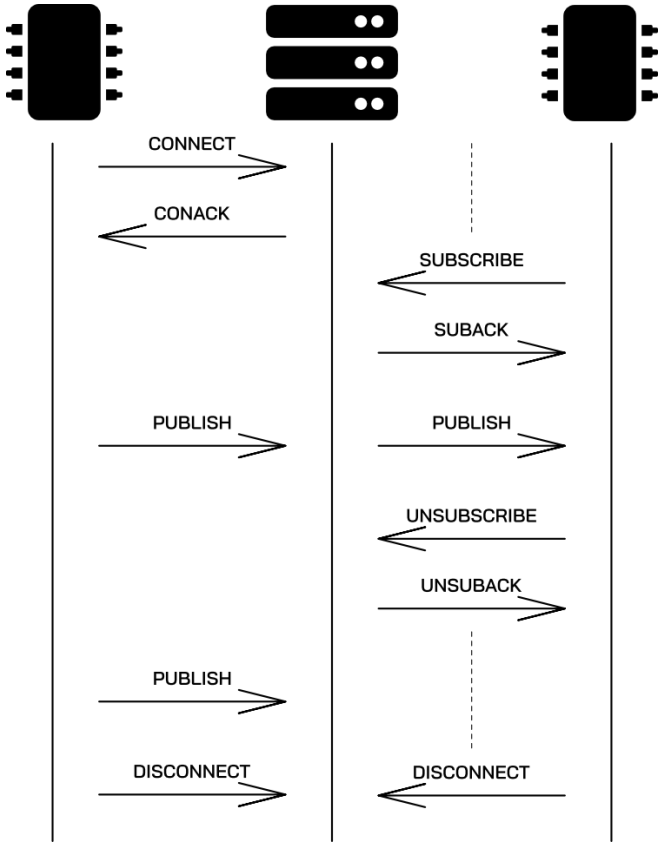
Ešte pred odoslaním prvej aplikačnej správy musí klient nadviazať spojenie s brokerom. Podľa dokumentácie [26] sa pre pripojenie používa štandardný port 1883 alebo 8883 pre zabezpečenú komunikáciu.

Po nadviazaní spojenia **??** pošle klient správu typu *CONNECT* brokeru. Pripojenie môže prebiehať i so základným zabezpečením pomocou mena a hesla. Broker následne odpovie správou typu *CONNACK*.

Od tohto momentu môžu nasledovať správy typu *SUBSCRIBE* alebo *PUBLISH* s názvom topicu, ktorý chce zariadenie odoberať, resp. publikovať. Broker tieto správy potvrdí správou typu *SUBACK*, resp. *PUBACK*. Potvrdenie o doručení publikovanej správy zariadenie obdrží len pri použití *QoS* vyššej alebo rovnej hodnote 1.

Zariadenie sa môže behom prevádzky odhlásiť z odberu niektorého topicu správou typu *UNSUBSCRIBE*, na ktorú broker odpovie správou typu *UNSUBACK*. Pri ukončení celej práce zariadenie posiela správu *DISCONNECT* a s brokerom už neudržiava spojenie.

V prípade, že sa klient odpojí, všetky správy z topicov, na ktoré bol prihlásený, mu budú po prihlásení opätovne doručené.



Obr. 3.2: Príklad komunikácie dvoch zariadení s brokerom pri kvalite služby 0. Všetky správy obsahujú rovnaký topic.

Zabezpečenie komunikácie

Množstvo pripojených zariadení a informácií prenesených po internete sa dennodenne zvyšuje, a aj preto sa bezpečnosť stáva čoraz dôležitejšiou. Cieľom MQTT je poskytnúť jednoduchý a ľahko použiteľný komunikačný protokol pre internet vecí. Samotný protokol špecifikuje niekoľko bezpečnostných mechanizmov.

Bezpečnostné mechanizmy sú vhodne popísané v článku publikovanom od spoločnosti HiveMQ [18], ktorá ich rozdeľuje do niekoľkých vrstiev. Každá vrstva zabraňuje rôznym druhom útokov. Aj keď je bezpečnosť zložitá, dáva zmysel stavať na všeobecne prijatých štandardoch.

Sieťová vrstva - Jedným zo spôsobov, ako zaručiť bezpečné a dôveryhodné pripojenie, je použitie fyzicky zabezpečenej siete alebo VPN (Virtual Private Network) pre všetkú komunikáciu medzi klientom a brokerom. VPN poskytuje istotu, že dáta sú prijímané len od autorizovaných klientov [18].
Transportná vrstva - Transportná vrstva a protokol TCP tvoria prevažnú časť zabezpečenia protokolu MQTT. Na tejto úrovni je možné použiť mechanizmus TLS, ktorý predstavuje bezpečný a osvedčený spôsob ako poskytnúť zabezpečený komunikačný kanál medzi klientom a serverom. TLS/SSL je kryptografický protokol, ktorý používa mechanizmus handshake k vytvoreniu zabezpečeného spojenia medzi klientom a brokerom. Po dokončení spojenia je nadviazaná šifrovaná komunikácia a žiadny útočník nemôže odpočúvať žiadnu časť komunikácie. Broker poskytuje digitálny certifikát X.509 vydaný dôveryhodnou autoritou, ktorý používajú klienti k overeniu identity brokera [18].
Aplikačná vrstva - Protokol MQTT poskytuje identifikátor klienta, používateľské meno a heslo na autentifikáciu zariadení na aplikačnej úrovni. Tieto vlastnosti poskytuje samotný protokol. Autorizáciu alebo kontrolu nad tým, čo môže každé zariadenie robiť, definuje konkrétna implementácia brokera [18].

Pri vynechaní zabezpečenia na transportnej vrstve sú dáta do brokeru odoslané v obyčajnom texte. Tým pádom sú prenášané informácie náchylné k odposluchu a útočníkom poskytujeme jednoduchý spôsob ako dáta prečítať, poprípade získať prihlasovacie údaje [26].

Kapitola 4

Prieskum cloudových služieb

Broker, ktorý zohráva pri komunikácii so zariadeniami hlavnú úlohu musí byť umiestnený na dostupnom mieste. Používateľ môže vybudovať svoje vlastné riešenie alebo môže využiť služby priamo určené pre Internet vecí od poskytovateľa cloudových služieb.

Služby od Amazon Web Services a Google Cloud môžu mať všestranné využitie alebo sú zamerané priamo na jedno technologické odvetvie. Spojením týchto služieb je možné vybudovať riešenie pre zariadenia internetu vecí, ktoré bude pokrývať všetky potreby užívateľa.

4.1 Cloudové služby od Amazon Web Services

Ako už bolo spomenuté v kapitole 2.1 služby sú poskytované rôznymi modelmi. Ja som sa zameral na tie najviac používané a na tie, ktoré môžu byť zaujímavé z pohľadu internetu vecí.

Amazon Elastic Compute Cloud

Skrátene Amazon EC2, je webová služba, ktorá poskytuje zákazníkom výpočtové zdroje v cloude. Jednoduché webové rozhranie služby Amazon EC2 umožňuje vytvoriť, konfigurovať a ovládať základné funkcie s minimálnym úsilím. Typicky ide o virtuálny počítač postavený na modeli *IaaS*, čím poskytuje úplnú kontrolu nad výpočtovými prostriedkami. Amazon EC2 ponúka platformu s výberom procesora, úložiska, siete a operačného systému [15].

AWS Lambda

Je serverless výpočtová služba, ktorá umožňuje spustiť kód bez poskytovania alebo správy serverov. Kód je spustený na základe prichádzajúcej žiadosti alebo udalosti z inej AWS služby, ktorému je automaticky pridelený výpočtový výkon. AWS Lambda funkcie je možné písať vo väčšine najznámejších jazykov, ako napríklad Node.js, Python, Go, Java a ďalšie [23].

Amazon Simple Storage Service

Skrátene Amazon S3, je služba na ukladanie súborov, ktorá poskytuje škálovateľnosť, zabezpečenie a výkon. Službu je možné použiť na ukladanie a ochranu dát pre rôzne prípady použitia, napríklad webové stránky, zálohovanie a bežné ukladanie [29].

Amazon Relational Database Service

Amazon RDS uľahčuje nastavenie, prevádzku a škálovanie relačnej databázy v cloude. Časovo náročné administratívne úlohy, ako je napríklad zabezpečenie hardvéru, nastavenie databázy, oprava a zálohovanie sú automatizované, čím sa potvrdzuje model *PaaS* z kapitoly 2.1. Užívateľ sa môže ihneď zamerať na tvorbu aplikácií. Medzi poskytované databázové servery jednoznačne patria PostgreSQL, MySQL, Oracle Database a SQL Server [28].

Amazon DynamoDB

Je NoSQL databázová služba, ktorá podporuje dátové štruktúry vo formáte key-value [14].

Amazon API Gateway

Hovoríme o plne spravovanej službe, ktorá uľahčuje vytváranie, publikovanie, monitorovanie a zabezpečenie API. Slúži ako vstupná brána do ďalších AWS služieb. Pomocou API Gateway je možné vytvárať RESTful API a WebSocket API [2].

Amazon CloudWatch

Táto služba je určená hlavne pre vývojárov a softvérových inžinierov, ktorá monitoruje a analyzuje správanie jednotlivých AWS služieb. Pomocou CloudWatch vieme rýchlejšie detekovať problémy, ktoré môžu nastať [9].

AWS Amplify

Sada nástrojov a služieb, ktoré môžu byť použité spoločne alebo samostatne, aby pomohli vývojárom webových a mobilných aplikácií vytvoriť škálovateľné aplikácie. Amplify podporuje populárne webové frameworky vrátane JavaScriptu, React, Angular, Vue a najznámejších mobilných platforiem Android a iOS [1].

AWS IoT Core

AWS IoT Core slúži ako vstupná bránna do AWS služieb určených pre Internet vecí. Umožňuje pripojiť zariadenia bez potreby poskytovania alebo správy serverov. Dokáže podporovať milióny zariadení a správ. Tieto správy dokáže spoľahlivo a bezpečne spracovávať a smerovať ďalším službám AWS alebo zariadeniam. V podstate je to broker, ktorý je spravovaný AWS. Pomocou webového rozhrania môže užívateľ spravovať zariadenia, certifikáty a mnoho ďalšieho [21].

Prostredníctvom Rules je zariadeniam umožnené komunikovať so službami AWS. Prichádzajúce správy sú filtrované podľa topicu, ktorý špecifikuje ako budú dáta spracované.

AWS IoT Analytics

Opäť hovoríme o plne spravovanej službe, ktorá uľahčuje spustenie a prevádzakovanie sofistikovaných analýz na prichádzajúcich dátach zo zariadení Internetu vecí. Pomocou strojového učenia a ďalších metód môžeme rozšíriť sadu údajov, ktoré budú uložené do databázy. Služba môže pomôct, napríklad poľnohospodárom aby pomohla určiť, kedy majú plodinu polievať, zavlažovať vinice, ktoré sú často obohatené údajmi zo senzora vlhkosti, čo umožňuje efektívnejšie využitie vody a maximalizovanie úrody [20].

AWS IoT Events

Služba umožňuje detekovať a reagovať na udalosti zo zariadení Internetu vecí [22]. Napríklad, pri detekovaní vysokej teploty v domácnosti je spustená akcia, resp. služba podľa preddefinovanej logiky.

Alexa Voice Service

Služba pomocou, ktorej je možné sa spojiť s virtuálnou hlasovou asistenkou Amazon Alexa a reagovať na prijaté inštrukcie. Veľký význam dáva pri integrácii s AWS IoT Core, kde je možné pomocou hlasu ovládať zariadenia [22].

AWS Identity and Access Management

Umožňuje bezpečne spravovať prístup k službám AWS. Týmto spôsobom umožníme pridať oprávnenia ďalšiemu užívateľovi k spravovaniu služieb vytvorených na inom účte. Alebo je možné vytvoriť roly, ktoré slúžia na bezpečné prepojenie dvoch služieb AWS [19]. Napríklad službe Lambda zabezpečíme prístup k informáciám zo služby DynamoDB.

Finančný plán[1]

Pre nového užívateľa je možné služby rozdeliť do troch kategórií:

Always free - Táto ponuka umožňuje zákazníkom používať služby zadarmo až do určených mesačných limitov tak dlho ako sú zákazníkmi AWS.
12 months free - Po registrácií do AWS je užívateľ automaticky zaradený do skupiny Free Tier na 12 mesiacov, ktorá ponúka vyskúšať si niektoré služby až do prekročenia limitov zdarma.
Trials - Ponuka na bezplatné vyskúšanie služby.

Sadzba za služby je hodinová, účtovaná vždy mesačne.

4.2 Cloudové služby od Google Cloud

Všetci poskytovatelia sa snažia dodávať užívateľovi čo najlepšie služby. A nie je to výnimkou ani pri Google Cloud. Základné služby sa neodlišujú od konkurencie, pretože cieľ použitia je rovnaký. Rozhodol som sa zamerať na služby, ktoré sa líšia od AWS.

Pub/Sub

Je to komunikačná služba, ktorá oddeľuje služby, ktoré vytvárajú udalosti od služieb, ktoré udalosti spracovávajú. Môže sa použiť ako vstupná brána pre ďalšie služby [27]. Čiastočne ju môžeme porovnať s AWS IoT Core Rules, ibaže tu má širšie zameranie a nemusí byť použitá len pri zariadeniach Internetu vecí.

App Engine

Služba postavená na modeli *PaaS*, ktorá umožňuje vývojárom hostovať webové aplikácie v datacentrách spravovaných spoločnosťou Google [3].

[1] https://aws.amazon.com/free/

BigQuery

Dátový sklad, ktorý spracováva a analyzuje veľké množstvo údajov. Do skladu je možné ukladať dáta z rôznych služieb a rovnako ich v prípade potreby načítať pre inú službu [4].

Dataflow

Služba určená na spracovávanie a streamovanie údajov v reálnom čase. Umožňuje vývojárom nastaviť kanály pre integráciu, prípravu a analýzu veľkého množstvo dát, a preto je často používaná so službou BigQuery [11].

Firebase

Platforma pre tvorbu mobilných aplikácií, ktorá zastrešuje viaceré aspekty životného cyklu produktu. Uľahčuje hlavne vývoj a následné monitorovanie aplikácie. Samotná platforma nie je súčasťou Google Cloud, ale využíva služby poskytované cloudom. Následne je možné pristupovať k obsahu služieb aj z Firebase aj z Google Cloud [17].

Finančný plán[2]

Spôsob účtovania poplatkov za využívanie služieb sa výrazne nelíši od iných poskytovateľov cloudových služieb. Sadzba je rovnako hodinová a účtovaná mesačne ako pri AWS. Taktiež existuje spôsob ako využívať niektoré služby zdarma.

Free Tier - Užívateľovi nie je účtovaný žiadny poplatok, pri používaní vybraných služieb do určených mesačných limitov.
Kredit $300 - Po registrácií do Google Cloud užívateľ automaticky obdrží kredit v hodnote $300, ktorý je platný po dobu maximálne 90 dní. Počas tejto doby môže užívateľ vyskúšať služby, ktoré nie sú súčasťou Free Tier, resp. môže vyskúšať ich pokročilejšie funkcie.

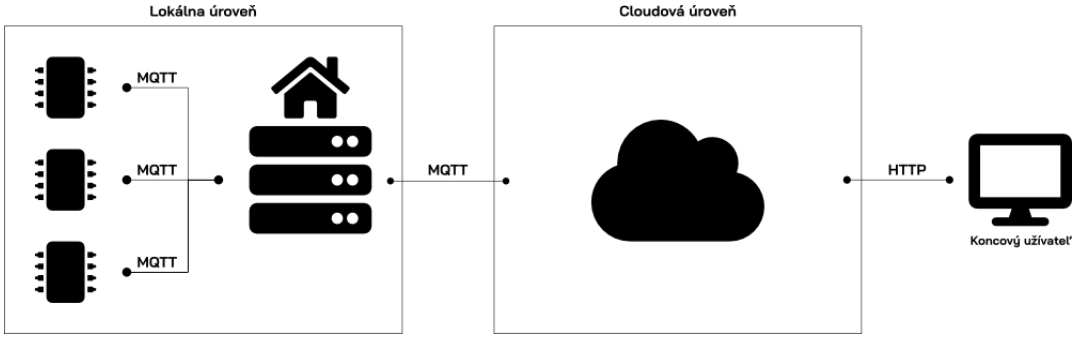
[2] https://cloud.google.com/free

Kapitola 5

Návrh riešenia

Existuje viac riešení ako prepojiť zariadenia internetu vecí s cloudom a aké služby použiť.

Rozhodol som sa navrhnúť riešenie, ktoré bude používať dva typy brokerov 5.1. Jeden lokálny, ktorý bude komunikovať so zariadeniami v domácnosti a druhý ako cloudová služba IoT Core, ktorá bude spracovávať dáta a inštrukcie zadané používateľom.



Obr. 5.1: Vizualizácia navrhnutého riešenia

5.1 Lokálna úroveň

Mikrokontrolér ako zariadenie v domácnosti bude spravovať rôzne senzory a ovládacie prvky. Snažil som sa vybrať také prvky, z ktorých bude možné informácie aj zbierať, ale aj ktorým bude možné inštrukcie posielať.

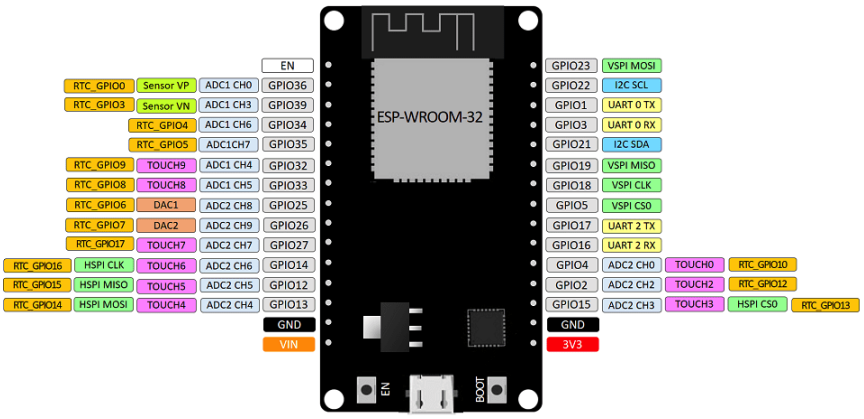
Takéto zariadenia budú priamo komunikovať s lokálnym brokerom, ktorý môže byť spustený na osobnom počítači alebo aj na Raspberry Pi[1]. Dôležité je, aby bol v dosahu zariadení.

Program, ktorý bude simulovať MQTT klienta može byť umiestnený na zariadení, ktoré nebude obsahovať senzory a ovládacie prvky. Údaje zo senzorov budú náhodne generované z vhodne zvoleného rozsahu. Následne pri monitorovaní zariadenia, bude možné v príkazovom riadku vidieť prijaté správy.

[1] https://www.raspberrypi.org

ESP32

Označenie pre sériu nízkonákladových a nízkoenergetických *System on a Chip* mikrokontrolérov s integráciou Wi-Fi a Bluetooth od spoločnosti Espresiff [16]. Sú vhodné práve pre IoT zariadenia, a preto som si zvolil práve vývojovú dosku ESP32-DEVKIT-V1 5.2[2], ktorá bude spravovať mnou zvolené ovládacie prvky.

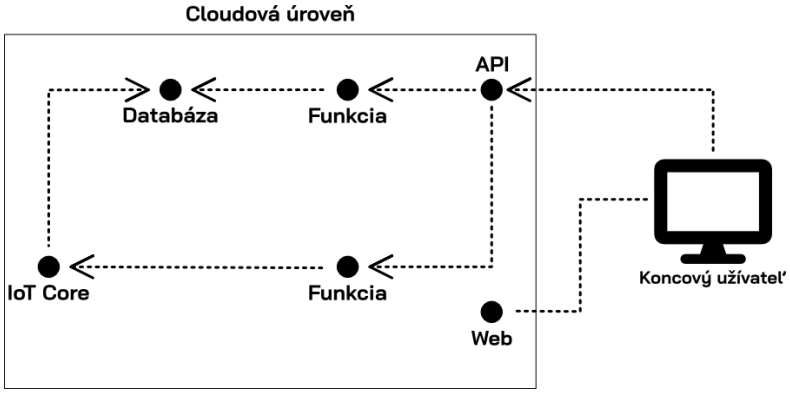


Obr. 5.2: Schéma portov zariadenia ESP32-DEVKIT-V1

5.2 Cloudová úroveň

Služba IoT Core v prepojení s lokálnym brokerom a spojením ďalších služieb od poskytovateľa vytvorí celý ekosystém. Prostredníctvom webovej aplikácie bude možné sledovať a ovládať zariadenia v domácnosti. Ďalšie služby budú spravovať údaje a reagovať na vonkajšie podnety, napríklad z webovej aplikácie viď 5.3.

Cloudová úroveň bude implementovaná samostatne pre AWS a samostante pre Google Cloud s tým, že prvky webovej aplikácie budú zachované v maximálnej možnej miere pre obe riešenia.



Obr. 5.3: Vizualizácia prepojenia cloudových služieb medzi sebou. Šípka znázorňuje tok požiadaviek.

[2] https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2018/08/ESP32-DOIT-DEVKIT-V1-Board-Pinout-30-GPIOs-Copy.png

Kapitola 6

Implementácia a vyhodnotenie

V tejto kapitole bude hovorené o použitých technológiách a implementácii navrhnutého riešenia, ktoré sa skladá z viacerých častí. Základ celej komunikácie je tvorený prostredníctvom protokolu MQTT. Postupne sa budem venovať témam od konfigurácie lokálneho brokera cez nastavenie MQTT klienta a implementáciu ovládacích prvkov pre zariadenia ESP32 až po správne nastavenie cloudových služieb.

6.1 Lokálna úroveň

Pod lokálnou úrovňou je myslená domácnosť a jej zariadenia komunikujúce prostredníctvom lokálnej Wi-Fi siete s lokálnym brokerom.

6.1.1 Konfigurácia brokera

Ako bolo spomenuté v predošlej kapitole, broker môže byť spustený aj na osobnom počítači. Keďže takéto riešenie najviac vyhovovalo mojim požiadavkám, rozhodol som sa ho použiť.

Jeden z najznámejších programov, ktorý zastupuje MQTT brokera sa volá Eclipse Mosquitto. Je open-source pod správou Eclipse Foundation a podporuje najnovšie verzie protokolu MQTT. Tento nenáročný broker je použiteľný pre všetky nízkoenergetické zariadenia, ale taktiež aj pre plnohodnotné servery [25].

Samotný Mosquitto broker bude spustený v Docker kontaineri. Docker je nástroj určený na uľahčenie tvorby, nasadenia a spúšťania aplikácií pomocou kontajnerov. Kontajner je štandardná jednotka softvéru, ktorá zabalí kód a všetky jeho závislosti do jedného balíčku nazývaného „image“, ktorý je samostatne spustiteľný nezávisle od operečného systému [13].

Bez použitia Dockeru by bolo nutné inštalovať spolu s brokerom aj ďalšie závislosti, ktoré by mohli zabrať viac času. Pomocou konfiguračného súboru *Dockerfile* alebo *docker-compose.yml* nastavím základne vlastnosti kontajneru. Je potrebné špecifikovať „image“, verziu, port a umiestnenie súborov, s ktorými bude broker pracovať.

Ja som zvolil najnovšiu verziu *‘eclipse-mosquitto:latest‘*, štandardný port 1883 určený pre MQTT a nasmeroval súbory a priečinky potrebné pre správne spustenie brokeru. Priečinky obsahujú konfiguračné a logovacie súbory, certifikáty, o ktorých bude popísané neskôr.

Konfiguračný súbor pre Mosquitto broker umožňuje prispôsobiť si broker svojím potrebám. Oficiálny repozitár pre Mosquitto obsahuje šablónu konfiguračného súboru[1] s veľmi vhodne vysvetelnými premennými a hodnotami, ktoré môže premenná nadobúdať.

[1] https://github.com/eclipse/mosquitto/blob/master/mosquitto.conf

Po nastavení všetkých premenných som spustil kontainer pomocou *'docker-compose up -d'*. Odteraz je zariadeniam umožnené sa pripojiť k lokálnemu brokeru.

6.1.2 Implementácia senzorov a ovládacích prvkov

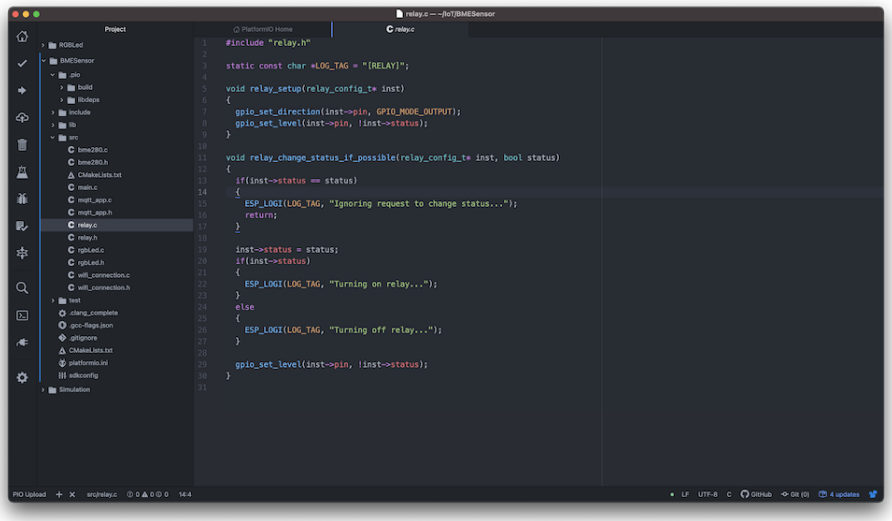
Pri implementácii akéhokoľvek produktu je možnosť zvoliť si vývojové frameworky. Mikrokontrolér ESP32 je taktiež možné naprogramovať v rôznych frameworkoch. Jednými z najznámejších je Espressif IDF (IoT Development Framework)[2] a Arduino IDE[3].

Napriek tomu, že celkový vývoj vyzerá jednoduhšie v Arduino IDE, rozhodol som sa použiť oficiálny framework Espressif IDF. Je open-source a poskytuje SDK pre akýkoľvek vývoj pomocou programovacích jazykov C a C++.

Máme na výber medzi manuálnou inštaláciou frameworku plus všetkých potrebných doplnkov a použitím zásuvného modulu pre vývojové prostredie. Ja som sa rozhodol pre druhú možnosť využiť nástroj PlatformIO, ktorý mi umožnil vyvíjať a spravovať zariadenie na jednom mieste.

PlatformIO[4]

Multiplatformový nástroj určený na vývoj, testovanie a monitorovanie aplikácií pre embedded systems. Vhodný aj pre vývoj zariadení Internetu vecí. Podporuje viac ako 1000 architektúr mikrokontrolérov a nechýba medzi nimi ani séria ESP32 s podporou frameworku Espressif IDF. Nástroj PlatformIO je možné integrovať do rôznych vývojových prostredí 6.1 a umožňuje nahrať vytvorený softvér priamo do zariadení.



Obr. 6.1: Vývojové prostredie Atom.io s integráciou nástroja PlatfromIO.

[2] https://www.espressif.com/en/products/sdks/esp-idf
[3] https://www.arduino.cc/en/software
[4] https://platformio.org

Pripojenie k Wi-Fi

Ešte pred pripojením zariadenia k brokeru je nutné nadviazať spojenie so samotnou Wi-Fi v domácnsoti. Dostupná knižnica *esp_wifi.h* od Espressif IDF[5] ponúka základné funkcie na pripojenie zariadenia k Wi-Fi. Súbory *wifi_connection.h* a *wifi_connection.c* rozširujú dostupnú knižnicu o metódy, ktoré nakonfigurujú zariadenie ESP32 podľa mojich požiadaviek.

Najprv je nutné zavolať metódu *wifi_initialize()*, ktorá nastaví základné vlastnosti a nadefinuje poslucháča pre zachytávanie správ. Po zavolaní metódy *wifi_connect()* 1 nadviažem spojenie s Wi-Fi pomocou zadaného mena a hesla.

```
const wifi_config_t wifi_config = {
    sta.ssid = "ssid",
    sta.password = "password123",
};

esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config);
esp_wifi_start();
```

Výpis 1: Pripojenie zariadenia k lokálnej Wi-Fi naprogramované v jazyku C.

Keďže pripojenie k Wi-Fi trvá niekoľko sekúnd je vhodné definovať metódu *callback*, ktorá bude zavolaná až po potvrdení, že pripojenie sa podarilo. V opačnom prípade môže nastať situácia, že funkcie, ktoré nasledovali po pripojení budú zavolané skôr ako bude samotné zariadenie pripojené k Wi-Fi.

Pripojenie k brokeru

Hlavná komunikácia klienta s brokerom je implementovaná v súbore *mqtt_app.c*. Rovnako ako pri implementácií Wi-Fi sa jedná o rozšírenie knižnice *mqtt_client.h* o metódy, ktoré uľahčujú pripojenie a spravujú prichádzajúce udalosti. Po úspešnom pripojení k Wi-Fi je zavolaná metóda *mqtt_app_start()* 2, ktorá má za úlohu nastaviť metódu pre spracovanie prichádzajúcich udalosti a pripojiť zariadenie k brokeru.

```
const esp_mqtt_client_config_t mqtt_config = {
    .uri = "mqtt://192.168.1.14",
};

esp_mqtt_client_handle_t c = esp_mqtt_client_init(&mqtt_config);
esp_mqtt_client_register_event(c, ESP_EVENT_ANY_ID, handler, c);
esp_mqtt_client_start(c);
```

Výpis 2: Pripojenie zariadenia k brokeru naprogramované v jazyku C.

[5] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html

Každá prijatá udalosť má svoj vlastný identifikátor, podľa ktorého sa systém rozhoduje, aká akcia bude nasledovať. Typ udalosti *MQTT_EVENT_SUBSCRIBED* označuje potvrdenie, že broker prijal aplikačnú správu na odber. Typ *MQTT_EVENT_PUBLISHED* označuje, že publikovaná správa bola prijatá brokerom. Tento typ udalosti môže byť zachytený len v prípade použitia $QoS >= 1$. Jednoducho povedané, klient zachytáva udalosti typu *SUBACK/PUBACK*, ktoré broker odoslal.

Jednou z najdôležitejších typov udalosti je jednoznačne *MQTT_EVENT_DATA*, ktorá obsahuje správu, o ktorú si klient zažiadal zaslaním správy typu *SUBSCRIBE*. Obsah správy, topic a dáta, je ďalej odovzdaný metóde podľa vopred nastaveného callbacku.

Implementácia teplotného senzoru

Pre zdieľanie údajov zo zariadenia Internetu vecí som si zvolil senzor teploty, tlaku a vlhkosti vzduchu BME280[6]. Ponúka I2C rozhranie, ktoré sa využíva pre komunikáciu medzi mikrokontrolérmi a senzormi. Dokáže spracovať teplotu od $-40°C$ do $85°C$ a tlak v rozmedzí $300 - 1100hPa$ pri pracovnom napätí $1.8V$ - $5V$.

Senzor obsahuje 4 piny, ktoré je potrebné prepojiť s ESP32 zariadením. Piny *GND* a *VIN* pre napájanie a piny *SDA*, a *SCL* pre komunikáciu so zariadením, ktoré je potrebné špecifikovať v dátovej štruktúre *bme280_config_t*[7].

Dáta zo senzorov bývajú zvyčajne odosielané v opakovaných intervaloch po celý čas svojho behu. Pomocou knižnice *esp_timer.h* a metódy *esp_timer_create()* je možné vytvoriť časovač, ktorý bude zavolaním metódy *esp_timer_start_periodic()* spustený. V 30 sekundových intervaloch je zavolaná funkcia na získanie údajov zo senzoru, ktoré sú následne publikované brokeru. Dáta sú odoslané vo formáte *JSON* 3 a s topicom *bt/{device_id}/sensor/bme*.

```
{
    "temperature": 22.6,
    "humidity": 47,
    "pressure": 958
}
```

Výpis 3: Príklad odoslaných dát zo senzoru BME280.

Pri implementácii som sa stretol s problémom pri volaní metódy *vTaskDelay()*, ktorá pôsobila dojmom časovača. Táto metóda zastavila chod celej aplikácie na určitý čas a MQTT klient nebol vôbec schopný prijímať správy od brokeru.

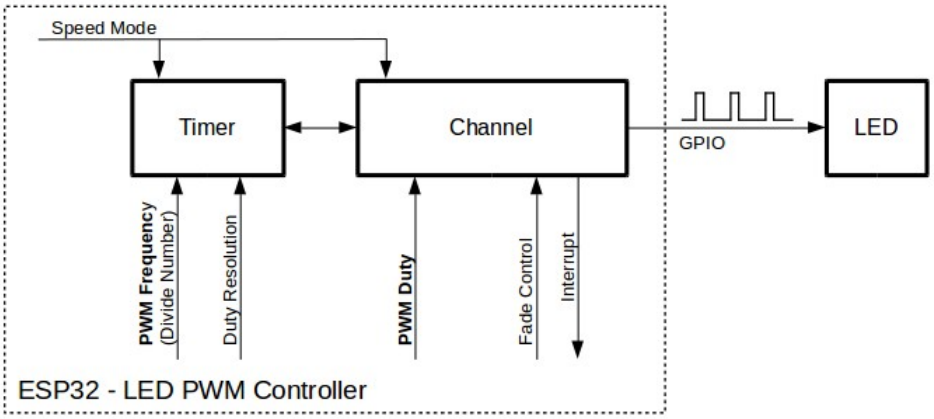
Implementácia LED RGB modulu

Jedným z vhodných ovládacích prvkov je LED RGB modul, ktorému je prostredníctvom pulznej šírkovej modulácie (PWM) možné meniť intezitu červenej, zelenej a modrej farby, čím je možné meniť celkovú výslednú farbu.

[6] https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/
[7] https://github.com/akurczyk/bme280_esp32_driver

Dostupná knižnica *driver/ledc.h* je primárne určená ovládať intenzitu LED, ale môže byť použitá aj na generovanie signálov PWM. Obsahuje 16 kanálov, ktoré môžu generovať nezávislé krivky, ktoré sa dajú použiť napríklad na riadenie RGB LED zariadení.

Komunikácia prebieha prostredníctvom portov *GPIO* 6.2[8], ktoré môžu byť použité ako vstupy alebo výstupy alebo oboje. Pre každú zložku farby je rezervovaný jeden port a kanál. Hodnota *duty_resolution* udáva počet bitov potrebných na pokrytie rozsahu hodnôt. Zložka RGB farby môže nadobúdať 256 hodnôt a tie dokážeme zaznamenať na 8 bitoch.



Obr. 6.2: Schéma komunikácie LED PWM ovládača s LED prostredníctvom GPIO

Keďže LED RGB modul je ovládaný užívateľom, musí byť prihlásený na odber. Prijaté dáta sú znovu vo formáte *JSON* 4 a s topicom *bt/{device_id}/led/rgb*.

```
{
    "R": 0,
    "G": 255,
    "B": 0
}
```

Výpis 4: Dáta prijaté od brokera indikujú, že LED RGB bude svietiť zelenou farbou.

Implementácia relé

Pomocou relé je možné zapínať, resp. vypínať zariadenia, ktoré potrebujú na svoj chod väčší prúd a napätie ako dokáže dodať ESP32 doska. Ja som sa rozhodol, že k relé pripojím LED žiarovku.

Pomocou dostupnnej knižnice *driver/gpio.h* a jednoduchej časti kódu je možné relé ovládať. Rovnako ako pri LED RGB moduli, aj pri relé prebieha komunikácia prostredníctvom portu *GPIO*. Keďže zariadenie ovláda relé, v kóde je potrebné nastaviť mód

[8] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/_images/ledc-api-settings.jpg

GPIO_MODE_OUTPUT, ktorý indikuje výstup signálu zo zariadenia. Výstup nadobúda dve hodnoty, 0 pre vypnutie a 1 pre zapnutie relé.

Pri implementácií sa nezabudlo na príjem opakovanej hodnoty, čiže ak je relé zapnuté a príde požiadavka ho znovu zapnúť, inštrukcia bude ignorovaná.

Rovnako ako LED RGB modul aj relé je ovládané užívateľom a musí byť prihlásené na odber správ. Správa ovladajúca relé prichádza vo vormáte *JSON* 5 s topicom *bt/{device_id}/bulb/switch*.

```
{
    "S": true
}
```

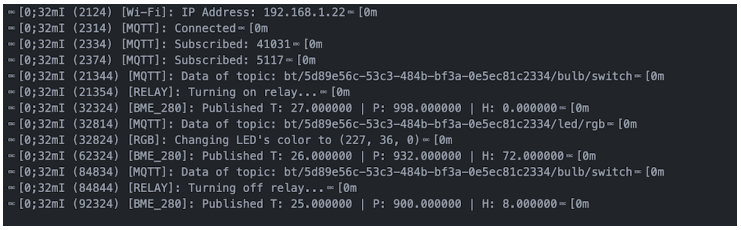
Výpis 5: Príklad správy, ktorá zopne relé a žiarovka začne svietiť.

6.1.3 Simulátor

Simulátor klienta je postavený na rovnakých základoch ako plnohodnotné zariadenia so senzormi a ovládacími prvkami. Zdrojový kód je potrebné nahrať do zariadenia ESP32, ku ktorému už nie sú pripojené ďalšie komponenty.

Zariadenie je po pripojení k WI-Fi a brokeru prihlásené na odber rovnakých topicov ako tomu bolo pri ovládacích prvkoch. Rozdiel je len v identifikátore zariadenia. V 30 sekundových intervaloch sú zo zariadenia odoslané dáta o teplote, tlaku a vlhkosti. Tieto dáta sú generované náhodne z vhodne zvoleného intervalu.

Pomocou vývojového nástroja PlatformIO je možné monitorovať 6.3 každú udalosť zo zariadenia.



Obr. 6.3: Výpis monitorovaných udalostí.

6.2 Cloudová úroveň

K lokálnemu brokeru je potrebné pripojiť ďalšie služby, ktoré umožnia užívateľovi spravovať ovládacie prvky a sledovať dáta zo senzorov. Ako bolo spomenuté v kapitole 4, cloudové služby ponúkajú možnosti ako spravovať zariadenia internetu vecí. Pomocou dostupných cloudových služieb od AWS a Google Cloud som nadviazal spojenie s lokálnym brokerom a prostredníctvom ďalších služieb som umožnil užívateľovi zariadenie v domácnosti ovládať.

6.2.1 Amazon Web Services

Základom celej komunikácie medzi lokálnym brokerom a cloudom je služba AWS IoT Core. V nasledujúcej časti je podrobnejšie popísaná integrácia cloudových služieb.

Prepojenie AWS IoT Core s lokálnym brokerom

Pre nadviazanie spojenia lokálneho brokera s AWS IoT Core je nutné použiť bezpečnostný mechanizmus TLS/SSL, ktorý pre svoj chod potrebuje vygenerovaný certifikát. Po vytvorení a stiahnutí certifikátu z AWS konzoly sa musí rozšíriť konfigurácia lokálneho brokera. Do Docker kontajnera boli presunuté súbory z certifikátu a následne do konfiguračného súboru brokera boli pridané ďalšie premenné, ktoré špecifikujú umiestnenie certifikátu a adresu cloudovej služby.

Týmto nastavením vzniklo spojenie typu bridge. Správa odoslaná zo zariadenia do lokálneho brokera je preposlaná do AWS IoT Core, kde je ďalej spracovaná. Rovnaké správanie môžeme zaznamenať aj opačného smeru, teda správa z ovládacieho panelu je prijatá najprv AWS IoT Core, ktorý ju prepošle do lokálneho brokera.

Ukladanie dát zo senzora BME280

Na ukladanie údajov zo senzora BME280 som sa rozhodol použiť službu DynamoDB. Po porovnaní s relačnou databázou mi prišlo vhodnejšie použiť NoSQL databázu, nakoľko zozbierané údaje môžeme zaradiť do oblasti big data.

Tabuľka DynamoDB

Vytvorená tabuľka *bt_bme_sensor_data* obsahuje tri stĺpce. Stĺpec *device_id* označovaný ako *Partition Key* bude obsahovať identifikátor zariadenia. Druhý stĺpec *sample_time* považovaný za *Sort key* obsahuje čas vykonania transakcie. Podľa *device_id* bude možné vyberať z databázy požadované údaje.

Namerané hodnoty zo senzora budú ukladané vo formáte *JSON* 6 do stĺpca *sensor_data*.

```
{
  "device_id": "cb084ee4-adba-495c-8d2c-2d5a00fe173b",
  "sample_time": 1616962348939,
  "sensor_data": {
    "humidity": 50,
    "pressure": 973,
    "temperature": 22.4
  }
}
```

Výpis 6: Príklad uložených dát v databáze DynamoDB.

Pravidlo pre ukladanie údajov do DynamoDB

V AWS IoT Core som vytvoril pravidlo, ktoré filtruje prijaté správy a spracuje len tie, ktoré vyhovujú zadanému topicu.

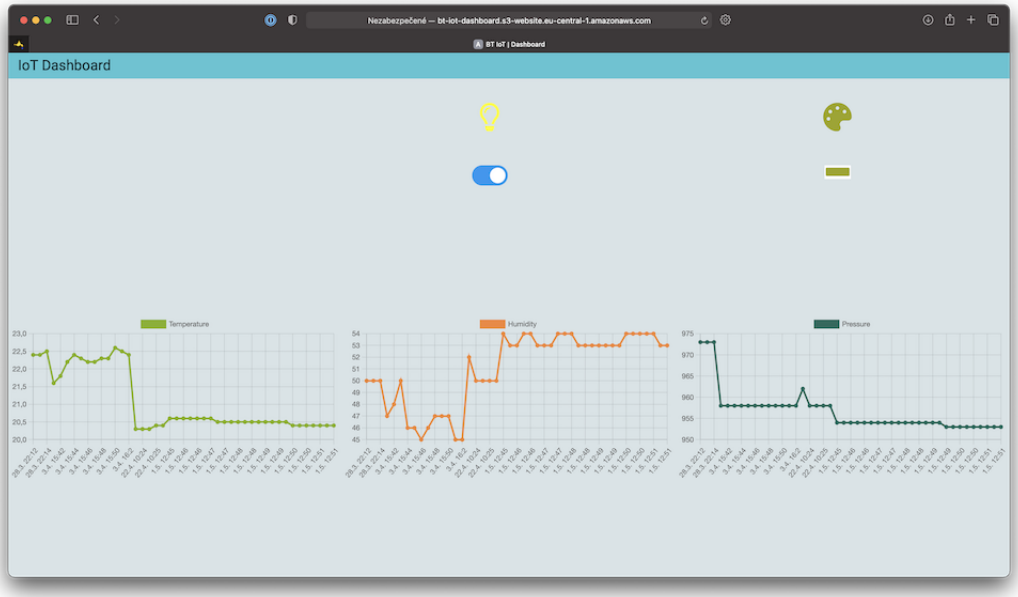
Pre výber požadovaných údajov a filtrovanie správ sa používa jazyk *SQL* 6.2.1, v ktorom je možné do topicu aplikovať zástupné znaky, ktoré uľahčia vyhľadávanie.

```
SELECT temperature, pressure, humidity FROM 'bt/+/sensor/bme'
```

Po vyselektovaní údajov je spustená preddefinovaná akcia, ktorá vytvorí záznam v tabuľke. Keďže potrebný identifikátor zariadenia sa nenachádza v údajoch, ale v topicu, je nutné ho získať odtiaľ. Topic sa skladá z úrovní oddelených lomkou a pomocou dostupnej funkcie *topic(int level)* viem vybrať priamo tú úroveň, ktorú potrebujem. Vybrané údaje zo správy budú uložené v stĺpci *sensor_data*.

Ovládací panel

Pomocou ovládacieho panelu je možné sledovať aktuálne hodnoty so senzoru a interagovať s relé a LED RGB modulom. Tento panel je riešený formou webovej stránky, ktorej súbory sú umiestnené v S3. Primárne, služba S3 slúži ako privátne úložisko alebo úložisko kam majú prístup len oprávené osoby alebo aplikácie. Je možné ju tiež použiť aj ako webový hosting pre statické stránky.



Obr. 6.4: Ovládací panel

Na vykresľovanie grafov bola použitá knižnica Chart.js[9] a komunikácia s AWS prebieha pomocou asynchronných volaní *ajax* z knižnice *jQuery*[10], kde obsah požiadavky aj odpovedi je vo formáte *JSON*.

API

Pre komunikáciu ovládacieho panela s AWS IoT Core je potrebné prejsť cez niekoľko vrstiev služieb.

Najprv som použil službu API Gateway, ktorá na základe navrhnutých endpointov rozhodne kam bude putovať prijatá požiadavka. Komunikácia s API Gateway je postavená na architektúre REST. Endpoint **PUT /device/state** slúži na aktuálizáciu stavu ovládacieho prvku. Pre získanie údajov o senzore som vytvoril endpoint **GET /device/{device_id}**.

Funkcie, ktoré ovládajú komponenty a získavajú údaje z databázy nie sú výpočtovo náročné a preto som sa rozhodol skúsiť použitie Lambda funkcií. Obe sú naprogramované v programovacom jazyku Java.

Lambda 1 - Funkcia, ktorá aktualizuje stav daných komponentov posiela správy priamo do AWS IoT Core. To označuje Lambda funkciu za MQTT klienta, ktorý komunikuje prostredníctvom protokolu MQTT. Pre túto potrebu bolo nutné vygenerovať nový certifikát a umiestniť ho k funkcii rovnako ako pri lokálnom brokeri. Prijatá požiadavka 7 je pred odoslaním do AWS IoT Core skontrolovaná, či neobsahuje neplatné údaje. Ak áno, funkcia a zároveň aj API vracia kód 422, ktorý označuje že prijatá požiadavka nemôže byť ďalej spracovaná. V opačnom prípade sa vráti kód 200, ktorý označuje úspešné spracovanie.

```
{
    "deviceId": "cb084ee4-adba-495c-8d2c-2d5a00fe173b",
    "topic": "/led/rgb",
    "params": {
        "R": 0,
        "G": 0,
        "B": 0
    }
}
{
    "deviceId": "cb084ee4-adba-495c-8d2c-2d5a00fe173b",
    "topic": "/bulb/switch",
    "params": {
        "S": false
    }
}
```

Výpis 7: Príklad requestov asynchrónne odoslaných z ovládacieho panelu do API Gateway a následne do Lambda funkcie.

[9] https://www.chartjs.org
[10] https://jquery.com

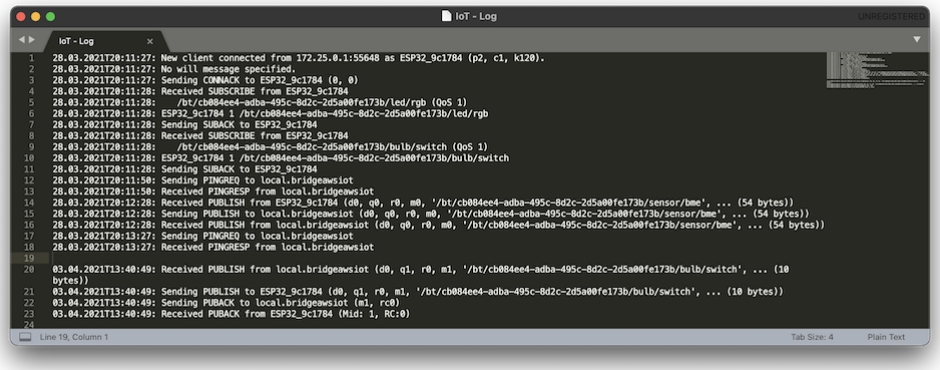
Lambda 2 - Funkcia získava údaje o senzore z databázy DynamoDB. Pre nadviazanie spojenia Lambda funkcie s DynamoDB bolo nutné vytvoriť alebo rozšíriť IAM rolu, ktorá povolí získavanie dát. Endpoint na získanie údajov obsahuje parameter *{device_id}*, ktorý pri zavolaní musí obsahovať identifikátor zariadenia. Podľa neho sa vyfiltrujú len tie dáta, ktoré potrebujem. Rovnako ako pri prvej Lambda funkcii, aj tu sa najprv ošetrí požiadavka a v prípade chyby bude vrátený kód 422. Pri úspechu je vrátený kód 200 spolu s obsahom 8 získaným z databázy.

```
{
    "data": [
        {
            "data": {
                "temperature": 22.4,
                "humidity": 50,
                "pressure": 973
            },
            "timestamp": 1616962348939
        }
    ]
}
```

Výpis 8: Získané data, ako odpoveď z API Gateway a Lambda funkcie.

Pri práci s Lambda funkciami, ktoré sú spustené po zavolaní požiadavky z ovládacieho panela som natrafil na problém, kedy prvé spustenie po dlhšej dobe zabralo príliš veľa času. Je to asi jedna z vlastností samotnej funkcie, keď systém uprednostní iné, častejšie volané, Lambda funkcie bežiace na rovnakom fyzickom stroji.

Na obrázku 6.5 je zobrazený záznam komunikácie medzi zariadením, lokálnym brokerom a cloudom.



Obr. 6.5: Záznam komunikácie medzi zariadením, lokálnym brokerom a cloudom.

6.2.2 Google Cloud

Rovnako ako pri AWS aj tu je základom celej komunikácie medzi lokálnym brokerom a cloudom služba IoT Core. Predtým ako som začal jednu zo služieb používať, bolo nutné v konzole Google Cloud vytvoriť projekt, do ktorého budú spadať mnou vybrané služby. V nasledujúcej časti je podrobnejšie popísaná integrácia cloudových služieb.

Prepojenie IoT Core s lokálnym brokerom

Komunikácia znovu používa spojenie typu bridge a vyžaduje použiť bezpečnostný mechanizmus TLS/SSL aby sa zabránilo podvodníkom pripojiť do IoT Core. Existujú dva typy certifikátov, kompletný a minimálny.

Kompletná sada umožňuje použitie naprieč všetkým Google Cloud službám. Zariadenia komunikujú priamo s MQTT serverom.

Minimálna sada je určená pre zariadenia s limitovanou pamäťou, ako sú mikrokontroléry. Zaručujú komunikáciu iba s IoT Core.

Po vytvorení a stiahnutí certifikátu je potrebné nakonfigurovať klienta MQTT, kde Google strikte určuje spôsob konfigurácie.

Ukladanie dát zo senzora BME280

Každá prijatá správa do IoT Core je distribuovaná službe Pub/Sub, ktorá danú správu spracuje.

Datastore

Služba Datastore umožňuje ukladať záznamy v NoSQL databáze [12], čo je pre údaje zo senzora vyhovujúce. Štruktúra tabuľky a jej obsah je takmer totožný s tabuľkou v DynamoDB. Keďže tu neexistuje *Partition Key* a *Sort Key*, ktorý by jednoznačne identifikoval záznam, je tu pridaný stĺpec ID, ktorý môžeme označiť za primárny kľúč a bude nadobúdať automatické hodnoty.

Funkcia pre ukladanie údajov do Datastore

Správa prijatá službou Pub/Sub je ďalej podľa nastavených pravidiel odoslaná do služby Cloud Function, ktorá vloží do Datastore záznam o nameraných hodnotách.

Ovládací panel

Ovládací panel, pomocou ktorého je možné sledovať aktuálne hodnoty zo senzoru je totožný s tým, ktorý bol použitý pri AWS. Jedinou zmenou sú API endpointy, ktoré tentokrát smerujú na služby od Google Cloud.

API

Pri použití služby API Gateway bolo trochu zložitejšie navrhnúť endpointy. Nie je možné ich vytvoriť pomocou webovej aplikácie, ale bolo nutné vytvoriť konfiguračný súbor a ten nahrať do služby API Gateway.

Súbor je definovaný OpenAPI[11] špecifikáciou, ktorá zahŕňa návrh API endpointov na architektúre REST. Návrh endpointu 9 obsahuje popis, parametre, či už v URL alebo v tele správy. Ďalej návratové kódy, podľa ktorých vývojár vie, či zavolaná funkcia prebehla v poriadku alebo nie. Google vyžaduje pridanie ďalších vlastných premenných, ktoré napríklad špecifikujú URL adresu funkcie.

```
"/device/{device_id}":
    get:
        description: "Get device data."
        operationId: "getDeviceDetail"
        x-google-backend:
            address: https://europe-west1-bt-iot-309217.cloudfunctions.net/...
            protocol: h2
        parameters:
        -
            name: device_id
            in: path
            required: true
            type: string
        responses:
            200:
                description: "Fetched data from Datastore."
                schema:
                    type: string
            422:
                description: "Invalid input."
```

Výpis 9: Príklad endpointu definovaného podľa štandardu OpenAPI

Funkcie, ktoré budú ovládať komponenty a získavať údaje z databázy opäť nie sú výpočtovo náročné, a preto som sa rozhodol skúsiť použitie služby Cloud Functions[12], ktorá je ekvivalentom služby AWS Lambda. Na naprogramovanie všetkých funkcií som použil programovací jazyk Python. Ich účel je úplne rovnaký ako som už spomenul pri Lambda funkciách od AWS. Prvá funkcia načítava údaje z databázy a druhá prijíma údaje z API Gateway a posiela ich ďalej.

Po napísaní zdrojového kódu, je nutné ho zostrojiť a nasadiť do Cloud funkcie. Tieto inštrukcie dokáže spracovať služba Cloud Build, ktorá ich vykoná na infraštrukúre poskytovateľa [7].

6.3 Porovnanie služieb

Cloudové služby od svojho vzniku radikálne vyspeli a výrazne rozšírili svoje možnosti. Na prvý pohľad sa môže zdať, že používanie služieb nie je zložité. Napriek tomu, človek potrebuje mať aspoň základné znalosti z vybranej oblasti a musí ešte venovať množstvo času pre získanie ďalších informácií ako služby fungujú.

Nielen z môjho pohľadu sú Amazon Web Services vyspelejšie ako Google Cloud. Ponúkajú prepracovanejšie služby, ktoré zabezpečia aj tým menej zdatným užívateľom pohodlnejšiu

[11] https://swagger.io/specification/
[12] https://cloud.google.com/functions

prácu. Napríklad konfigurácia API endpointov bola možná aj pomocou webového rozhrania, pričom na Google Cloud bolo nutné použiť výhradne súbor s OpenAPI špecifikáciou. Služba IoT Core v kombinácií s lokálnym brokerom vytvárajúca bridge je oveľa flexibilnejšia pri AWS. Google Cloud definuje celkové pripojenie a názvy topicov.

Na druhej strane Google Cloud ponúka intuitívnejšie a jednotnejšie webové rozhranie oproti Amazon Web Services. Platforma taktiež podporuje použitie viacerých programovacích jazykov pri tvorbe Cloud Functions vo webovej aplikácii.

Vysoko nastavená latka už dnes neumožňuje novovznikajúcim spoločnostiam zameraných na cloud computing preraziť na trhu. Je výhodnejšie sa zamerať na vytvorenie produktu, ktorý bude postavený na cloudových službách. Každopádne, obaja poskytovatelia ponúkajú takmer totožné služby a závisí iba od našich požiadaviek, ktorého si vyberieme.

Kapitola 7

Záver

Cieľom tejto práce bolo preskúmať a následne porovnať jadných z najväčších poskytovateľov cloudových služieb Amazon a Google so zameraním na problematiku služieb inteligentnej domácnosti. Po preskúmaní ponúkaných služieb a ujasnení si dovtedy neznámych odvetví som bol schopný navrhnúť riešenie, ktoré prepojí zariadenia v domácnosti s existujúcimi cloudovými službami.

Riešenie zahŕňa samostatnú implementáciu pre Amazon Web Services a taktiež pre Google Cloud. Simulátor Smart Home zariadení musí byť spustený na mikrokontroléri ESP32, ku ktorému nie sú pripojené ovládacie prvky a senzory, kde údaje sú generované náhodne. Po oboznámení sa so službami, môžem povedať, že služby od AWS ponúkajú väčšie možnosti ako ich použiť. Google je z môjho pohľadu skôr konzervatívny, strikte definuje ako sa musia služby používať, čo môže byť pre niekoho obmedzujúce. Napriek môjmu úsiliu pri implementácii Google služieb sa mi nepodarilo prepojiť lokálnu úroveň s cloudovou.

Práca mi umožnila použiť cloudové služby, ktoré sú v dnešnej dobe veľmi používané a rozšírili môj rozhľad. Výsledkom celej práce je kompletný balík umožňujúci používať v domácnosti inteligentné zariadenia pripojené k lokálnemu brokeru, ktorý je nakonfigurovaný ako bridge a komunikuje so službou IoT Core v cloude. Prepojením dostupných cloudových služieb vzniká produkt ovládacieho panela, pomocou ktorého je možné sledovať a ovládať zariadenia v domácnosti.

V budúcnosti by stálo za zváženie použiť komunikáciu zariadenia priamo so službou IoT Core, bez využitia lokálneho brokera nakonfigurovaného ako bridge.

Literatúra

[1] Amazon Web Services, I. *AWS Amplify* [online]. Dostupné z: https://aws.amazon.com/amplify/.

[2] Amazon Web Services, I. *Amazon API Gateway* [online]. Dostupné z: https://aws.amazon.com/api-gateway/.

[3] Google, I. *App Engine* [online]. Dostupné z: https://cloud.google.com/appengine.

[4] Google, I. *BigQuery* [online]. Dostupné z: https://cloud.google.com/bigquery.

[5] Upadhyay, I. *Top 10 Major Characteristics of Cloud Computing* [online]. 2021. Dostupné z: https://www.jigsawacademy.com/blogs/cloud-computing/characteristics-of-cloud-computing/.

[6] UMB. *Cloud computing* [online]. Dostupné z: https://lms.umb.sk/pluginfile.php/171114/mod_resource/content/2/kap2.pdf.

[7] Google, I. *Cloud Build* [online]. Dostupné z: https://cloud.google.com/build.

[8] Amazon Web Services, I. *Types of Cloud Computing* [online]. Dostupné z: https://aws.amazon.com/types-of-cloud-computing/.

[9] Amazon Web Services, I. *Amazon CloudWatch* [online]. Dostupné z: https://aws.amazon.com/cloudwatch/.

[10] WebSupport. *Čo je to Cloud a Cloud computing?* [online]. Dostupné z: https://www.websupport.sk/faq/co-je-to-cloud-a-cloud-computing.

[11] Google, I. *Dataflow* [online]. Dostupné z: https://cloud.google.com/dataflow.

[12] Google, I. *Datastore* [online]. Dostupné z: https://cloud.google.com/datastore.

[13] Docker. *What is a Container?* [online]. Dostupné z: https://www.docker.com/resources/what-container.

[14] Amazon Web Services, I. *Amazon DynamoDB* [online]. Dostupné z: https://aws.amazon.com/dynamodb/.

[15] Amazon Web Services, I. *Amazon Elastic Compute Cloud* [online]. Dostupné z: https://aws.amazon.com/ec2/.

[16] SYSTEMS, E. *ESP32* [online]. Dostupné z: https://www.espressif.com/en/products/socs/esp32.

[17] Google, I. *Firebase* [online]. Dostupné z: https://firebase.google.com.

[18] HiveMQ. *MQTT Security Fundamentals* [online]. Apríl 2015. Dostupné z: https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals/.

[19] Amazon Web Services, I. *AWS Identity and Access Management* [online]. Dostupné z: https://aws.amazon.com/iam/.

[20] Amazon Web Services, I. *AWS IoT Analytics* [online]. Dostupné z: https://aws.amazon.com/iot-analytics/.

[21] Amazon Web Services, I. *AWS IoT Core* [online]. Dostupné z: https://aws.amazon.com/iot-core/.

[22] Amazon Web Services, I. *What is AWS IoT Events?* [online]. Dostupné z: https://docs.aws.amazon.com/iotevents/latest/developerguide/what-is-iotevents.html.

[23] Amazon Web Services, I. *AWS Lambda* [online]. Dostupné z: https://aws.amazon.com/lambda/.

[24] Miklašová, K. *Smart Home - Umelá inteligencia v domácnosti* [online]. Masarykova univerzita, Fakulta informatiky, 2016. Dostupné z: https://nlp.fi.muni.cz/uui/referaty2016/kristina_miklasova/referat.pdf.

[25] Eclipse. *Eclipse Mosquitto* [online]. Dostupné z: https://mosquitto.org.

[26] OASIS. *MQTT Version 5.0* [online]. Editovali Andrew Banks, Ed Briggs, Ken Borgendale, a Rahul Gupta, 7. marca 2019. Dostupné z: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[27] Google, I. *Pub/Sub* [online]. Dostupné z: https://cloud.google.com/pubsub.

[28] Amazon Web Services, I. *Amazon Relational Database Service* [online]. Dostupné z: https://aws.amazon.com/rds/.

[29] Amazon Web Services, I. *Amazon Simple Storage Service* [online]. Dostupné z: https://aws.amazon.com/s3/.

[30] Weber, R. H. a Weber, R. *Internet of Things: Legal Perspectives*. 1. vyd. Springer-Verlag Berlin Heidelberg, 2010. ISBN 978-3-642-11710-7.

[31] Zelenay, M. *Výzvy testovania internetu vecí*. Brno, 2016. Diplomová práca. Masarykova univerzita, Fakulta informatiky.

Príloha A

Obsah priloženého pamäťového média

Na priloženom pamäťovom médiu sa nachádzajú tieto priečinky a súbory:

- **/ESP32/** - zložka obsahujúca zdrojové kódy nahrávané do zariadenia ESP32
  - **BME_LED_RELAY/** - zložka so zdrojovými kódmi, ktoré podporujú reálne komponenty
  - **Simulation/** - zložka so zdrojovými kódmi, kde je chovanie komponent simulované
- **/LocalBroker/Mosquitto/** - zložka obsahujúca konfiguračné súbory pre spustenie lokálneho brokera
- **/Cloud/** - zložka obsahujúca zdrojové súbory a screenshoty použité pri cloudových službách
- **/Thesis/** - zdrojové súbory a výsledný text bakalárskej práce

Príloha B

Manuál

B.1 Spustenie lokálneho brokera

Pre spustenie lokálneho brokera na osobnom počítači je nutné mať nainštalovaný program Docker a Docker Compose.

V priečinku */LocalBroker/Mosquitto/* sa nachádza súbor *docker-compose.yml*, ktorý je potrebné spustiť pomocou príkazu *'docker-compose up -d'*. Priečinok *data-mosquitto* sa musí nachádzať v rovnakej zložke ako konfiguračný súbor kontainera.

B.2 Spustenie zariadenia ESP32

Pre nahranie súborov do zariadenia ESP32 je potrebný framework *PlatformIO* integrovaný do *Atom.io* alebo *Visual Studio Code* vývojového prostredia.

Po vytvorení nového projektu a zvolení frameworku Espressif IDF je potrebné presunúť obsah priečinku */ESP32/BME_LED_RELAY/* alebo */ESP32/Simulation/* do projektu. V súbore *src/main.c* je ešte nutné definovať meno a heslo dostupnej Wi-Fi a IP adresu lokálneho brokera. Následne je možné projekt zostrojiť a nahrať do zariadenia.

Ovládací panel sa nachádza na stránkach:

- **Amazon** - `http://bt-iot-dashboard.s3-website.eu-central-1.amazonaws.com`
- **Google** - `https://storage.googleapis.com/bt-iot-dashboard/index.html`